



PATENT ABSTRACTS OF JAPAN

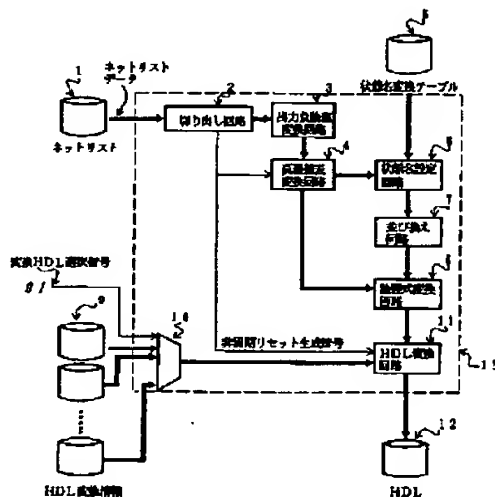
(11) Publication number: **08115357 A**(43) Date of publication of application: **07.05.96**(51) Int. Cl. **G06F 17/50**(21) Application number: **06253510**(71) Applicant: **FUJITSU LTD**(22) Date of filing: **19.10.94**(72) Inventor: **IGUCHI KATSUMI**(54) **METHOD AND DEVICE FOR CONVERSION OF NET LIST INTO HARDWARE LANGUAGE**

(57) Abstract:

PURPOSE: To convert held net list resources into the hardware description language(HDL) by converting a combinational circuit into a truth table, expanding the ignorance logic of the current state code and next state code of the truth table, and generating the HDL from the truth table containing state names.

CONSTITUTION: A truth value table converting circuit 4 converts the combinational circuit 30 after the negative logic output of a flip-flop part was removed by an output negative logic converting circuit 3 into a truth value table. Then, the ignorance logic of the current state code and next state code is expanded. In the expanded truth value table, temporary state name is assigned to a state code. A rearranging circuit 7 sorts the truth value table wherein the temporary name is set with the current state name to put codes together by the current state name. When an asynchronous reset generation signal is inputted from a segmentation circuit 2 to the HDL converting circuit 11, the HDL is generated from an asynchronous reset conditional expression on the basis of hardware conversion information corresponding to a conversion HDL select signal 91. Further, the HDL is generated from the truth value table containing the state names and the processing is ended.

COPYRIGHT: (C)1996,JPO



(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平8-115357

(43) 公開日 平成8年(1996)5月7日

(51) Int.Cl.⁶

G 0 6 F 17/50

識別記号

庁内整理番号

F I

技術表示箇所

9191-5H

9191-5H

G 0 6 F 15/ 60

6 5 4 A

6 5 4 K

審査請求 未請求 請求項の数 2 O L (全 9 頁)

(21) 出願番号

特願平6-253510

(22) 出願日

平成6年(1994)10月19日

(71) 出願人 000005223

富士通株式会社

神奈川県川崎市中原区上小田中1015番地

(72) 発明者 井口 克己

神奈川県川崎市中原区上小田中1015番地

富士通株式会社内

(74) 代理人 弁理士 林 恒徳

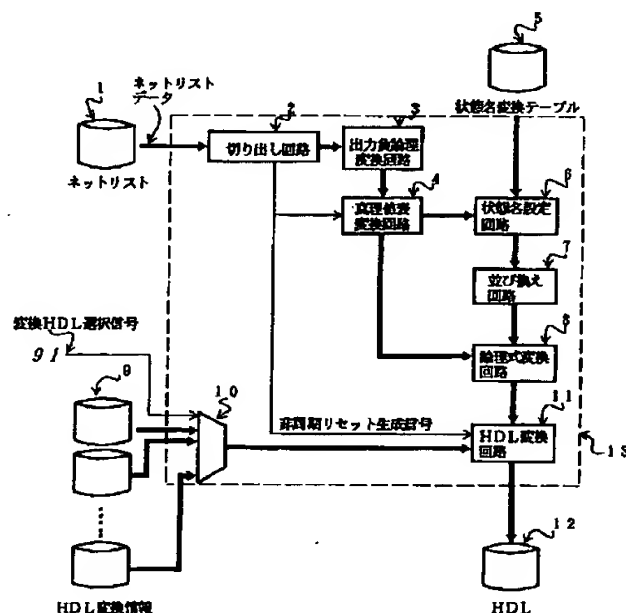
(54) 【発明の名称】 ネットリストのハードウェア言語への変換方法及び装置

(57) 【要約】

【目的】 状態遷移回路に基づく接続情報がリストされたネットリストのハードウェア記述言語への変換方法を提供する。

【構成】 フリップフロップで構成されるレジスタ部分とレジスタ部分を制御する組み合わせ回路部分を有する状態遷移回路に基づく接続情報がリストされたネットリストからフリップフロップ部分と組み合わせ回路部分を切り出し、フリップフロップ部分の負論理出力を該組み合わせ回路部分から取り除いて組み合わせ回路を真理値表に変換し、真理値表の現状態コード、次状態コードの無視論理を展開し、現状態コード、次状態コードを仮りの状態名に置き換え、現状態名毎に該真理値表をまとめ、次いで、状態名を含んだ真理値表からハードウェア記述言語を生成する。

実施例構成ブロック図



【特許請求の範囲】

【請求項 1】 フリップフロップで構成されるレジスタ部分と該レジスタ部分を制御する組み合わせ回路部分を有する状態遷移回路に基づく接続情報がリストされたネットリストから該フリップフロップ部分と組み合わせ回路部分を切り出し、

該フリップフロップ部分の負論理出力を該組み合わせ回路部分から取り除いて該組み合わせ回路を真理値表に変換し、

該真理値表の現状態コード、次状態コードの無視論理を展開し、

該現状態コード、次状態コードを仮りの状態名に置き換え、

現状態名毎に該真理値表をまとめ、

次いで、状態名を含んだ真理値表からハードウェア記述言語を生成するようにしたことを特徴とするネットリストのハードウェア記述言語への変換方法。

【請求項 2】 フリップフロップで構成されるレジスタ部分と該レジスタ部分を制御する組み合わせ回路部分を有する状態遷移回路に基づく接続情報がリストされたネットリストを記憶する第一のメモリと、

状態名変換テーブルを記憶する第二のメモリと、

特定のハードウェア記述言語への変換情報を記憶する第三のメモリと、

演算処理装置を有し、

該演算処理装置は、該第一のメモリからのネットリストから該フリップフロップ部分と組み合わせ回路部分を切り出す切り出し、

該フリップフロップ部分の負論理出力を該組み合わせ回路部分から取り除いて該組み合わせ回路を真理値表に変換し、

該真理値表の現状態コード、次状態コードの無視論理を展開し、

該現状態コード、次状態コードを該第二のメモリからの状態名変換テーブルに基づき、仮りの状態名に置き換え、

現状態名毎に該真理値表をまとめ、

次いで、状態名を含んだ真理値表から該第三のメモリからの特定のハードウェア記述言語への変換情報に基づき、ハードウェア記述言語を生成するように構成されたことを特徴とするネットリストのハードウェア記述言語への変換装置。

【発明の詳細な説明】

【0001】

【産業上の利用分野】 本発明は、ネットリストのハードウェア記述言語への変換方法及び装置に関し、特に、フリップフロップで構成されるレジスタ部分と該レジスタ部分を制御する組み合わせ回路部分を有する状態遷移回路に基づく接続情報がリストされたネットリストのハードウェア記述言語への変換方法及び装置に関する。

【0002】

【従来の技術】 従来回路設計を行うに際し回路図を直接手で書いたり、或いは CAD (Computer Aided Design) システムを用いずに、電気的特性に依存しない機能マクロを用いた機能図を使って回路設計を行う場合がある。

【0003】 即ち、設計者は、この機能図を基にして回路の機能検証を行う機能シミュレーションや実際のセルにマッピングを行う回路合成を行って、目的とする回路を設計していた。

10 【0004】 しかし、機能図のみでは、目的とする回路の機能を十分に表現出来ない。このために回路の動作をソフトウェアのプログラミング言語のように柔軟に記述できるようにし、回路設計用のプログラミング言語で回路を設計することが、現在の回路設計の主流になりつつある。

【0005】 一方、この回路設計用のプログラミング言語は、ハードウェア記述言語 (HDL) と呼ばれ、世界的に広く使用されているハードウェア記述言語 (HDL) として、VHDL あるいは、Verilog-HDL と呼ばれるものがある。

20 【0006】 市販されているシュミレータや回路合成システムもこれらのハードウェア記述言語 (HDL) を入力とするものが多く有る。したがって、市販の CAD システムを使用する時やさまざまな CAD システムとインタフェースをとる場合には、ハードウェア記述言語 (HDL) を経由して設計データを CAD システムに入力する必要が生じる。

【0007】 しかし、かかるハードウェア記述言語 (HDL) ではない、接続情報のリストであるネットリストを資産として保有する場合は、これまでは、市販の CAD システムを使用しようとする場合、そのままでは使用出来ないことになる。

【0008】 したがって、ネットリストをハードウェア記述言語 (HDL) に変換する必要がある。この変換に際し、セルで記述されたランダムロジックは、対応する論理をそのままハードウェア記述言語 (HDL) の論理式に変換すればよい。

40 【0009】 しかし、フリップフロップ (FF) で構成された制御部分とランダムロジックで構成された組み合わせ回路部分を持つような状態遷移回路については、ハードウェア記述言語 (HDL) に変換する場合、割りつけられた状態コードを見つけ出す技術が必要であった。

【0010】 そして、この割りつけられた状態コードを見つけ出す技術として、これまで適当な方法がなく、したがって、保有するネットリスト資産を市販の CAD システムにおいて使用することが出来ないと言う問題が存在した。

【0011】

50 【発明が解決しようとする課題】 したがって、本発明の目的は、保有するネットリスト資産をハードウェア記述

言語（HDL）に変換するハードウェア記述言語への変換方法及び装置を提供することにある。

【0012】更に、本発明の目的は、特にフリップフロップ（FF）で構成された制御部分とランダムロジックで構成された組み合わせ回路部分を持つような状態遷移回路についてのネットリスト資産をハードウェア記述言語（HDL）に変換するハードウェア記述言語への変換方法及び装置を提供することにある。

【0013】また、本発明の目的は、ハードウェア記述言語に依存しないで、保有するネットリスト資産をハードウェア記述言語（HDL）に変換するハードウェア記述言語への変換方法及び装置を提供することにある。

【0014】

【課題を解決するための手段及び作用】本発明にしたがうネットリストのハードウェア記述言語への変換方法は、まずフリップフロップで構成されるレジスタ部分とこのレジスタ部分を制御する組み合わせ回路部分を有する状態遷移回路に基づく接続情報がリストされたネットリストからフリップフロップ部分と組み合わせ回路部分を切り出す。

【0015】次いで、フリップフロップ部分の負論理出力を組み合わせ回路部分から取り除いて、組み合わせ回路を真理値表に変換し、この真理値表の現状態コード、次状態コードのdon't care（無視論理）を展開し、現状態コード、次状態コードを仮りの状態名に置き換え、現状態名毎に真理値表をまとめる。

【0016】次いで、状態名を含んだ真理値表からハードウェア記述言語を生成するようにする。

【0017】更に、本発明にしたがうネットリストのハードウェア記述言語への変換装置は、フリップフロップで構成されるレジスタ部分とこのレジスタ部分を制御する組み合わせ回路部分を有する状態遷移回路に基づく接続情報がリストされたネットリストを記憶する第一のメモリと、状態名変換テーブルを記憶する第二のメモリと、特定のハードウェア記述言語への変換情報を記憶する第三のメモリと、演算処理装置を有する。

【0018】そして、演算処理装置は、第一のメモリからのネットリストからフリップフロップ部分と組み合わせ回路部分を切り出す切り出す。フリップフロップ部分の負論理出力を組み合わせ回路部分から取り除いて、組み合わせ回路を真理値表に変換する。

【0019】そして、真理値表の現状態コード、次状態コードのdon't care（無視論理）を展開し、現状態コード、次状態コードを該第二のメモリからの状態名変換テーブルに基づき、仮りの状態名に置き換え、現状態名毎に該真理値表をまとめる。

【0020】次いで、状態名を含んだ真理値表から第三のメモリからの特定のハードウェア記述言語への変換情報に基づき、ハードウェア記述言語を生成するように構成される。

【0021】

【実施例】図1は、本発明にしたがうネットリストのハードウェア記述言語への変換装置の実施例ブロック図である。更に図2は、図1に対応する実施例処理フローである。

【0022】図1において、1は、状態遷移回路の接続情報を記述するネットリストを記憶するメモリ装置である。13は、本発明にしたがうネットリストのハードウェア記述言語への変換方法を実行制御する装置であり、10、演算処理装置によって構成される。

【0023】この変換方法を実行制御する装置13内には、各実行制御の過程の機能を回路部分として示し以下、回路として説明するが、ハード回路部分として実現することも、或いは対応する回路機能をソフトウェア処理により実行することも可能である。

【0024】5は、後に説明するネットリストのハードウェア記述言語への変換方法の実行の際に真理値表に状態名を設定する際の状態名を記述する状態名変換テーブルを記憶するメモリである。

【0025】9は、複数のHDL変換情報であり、例えば既述したVHDL、Verilog-HDL等のハードウェア記述言語（HDL）の変換情報を記憶するメモリである。この内、オペレータの操作により変換HDL信号91を入力して、変換を行うべき対応するハードウェア記述言語（HDL）の変換情報が選択される。

【0026】12は、変換されたハードウェア記述言語（HDL）を記憶するメモリ装置である。

【0027】次に変換方法を実行制御する装置13における変換の実際を以下に説明する。

【0028】図2の動作フローにおいて、変換処理がスタートすると先ず図1の切り出し回路2において、記憶装置1のネットリストから状態遷移回路のフリップフロップ部分と組み合わせ回路の切り出しを行う（ステップS1）。

【0029】即ち、レジスタ（フリップフロップFF0～FFn部分）とそれを制御する組み合わせ回路部分30により構成される状態遷移回路が図3に示すように一般的に示される。このような状態遷移回路がメモリ装置1に記憶されるネットリストから切り出される。

【0030】この時、非同期リセット信号の有無を判断する（ステップS2）。非同期リセット信号がある場合は、非同期のリセット条件式を求める（ステップS3）。この非同期リセット信号の有無の判断及び非同期のリセット条件式を求める処理も切り出し回路2において行われる。

【0031】次に切り出された状態遷移回路の内、フリップフロップ（FF）部分で負論理の出力QNが着目している組み合わせ部分30に接続している場合、出力負論理変換回路3によりFFの負論理出力を組み合わせ回路から取り除く処理を行う（ステップS4）。

【0032】即ち、図4に示されるように、FF部分で負論理の出力QNnに着目している組み合わせ部分30に接続している場合〔図4（1）〕、Qnにインバータ40を挿入し、出力QNnを使用しないように変更する〔図4（2）〕。

【0033】この段階で、図5に示すように、FF部分で負論理の出力QNnを使用しないレジスタ部分（FF0～FFn）と組み合わせ回路30の状態遷移回路ブロック図を得る。

【0034】次いで、真理値表変換回路4において、組み合わせ回路30を真理値表に変換する（ステップS5）。この時、真理値表変換回路4には、先に求めた非同期のリセット条件式（非同期リセット生成信号）が入力される。

【0035】真理値表変換回路4により変換された結果は、図6に示す如くである。図6において、A0で示される入力データ及び対応する現状態コードは、1または0のいずれも取りうるdon't care（無視論理）部分である。

【0036】これを現状態コード、次状態コードのdon't care（無視論理）を展開する（ステップS6）。この展開された状態が図7に示される。図6のA0の部分が発7でA1、A2に展開されている。

【0037】次いで、図7に展開された真理値表に対し、状態コードに仮の状態名を割り当てる（ステップS7）。この仮の状態名の割り当ては、任意に設定出来るが、仮の状態名として、メモリ5に記憶されている状態名変換テーブル5に登録されているものが使用される。

【0038】即ち、図1の状態名設定回路6において、メモリ5に記憶されている状態名変換テーブルの状態名を用いて任意に割り当てが行われる。この結果例えば、図8のように入力データ、出力データに対して仮の状態名ST0～ST3等が設定される。

【0039】仮の状態名ST0～ST3等が設定された真理値表に対し、並べ換え回路7により、更に現状態名でソートし、現状態名毎にまとめが行われる（ステップS8）。このようにしてまとめられた真理値表は、図9に示される如くなる。

【0040】したがって、この図9に対応する状態遷移図が図10に示される。例えば、図10において、状態ST0の時、入力データが1...11であれば、次の状態は、状態ST1となり、入力データが0...1-であれば、次の状態は、状態ST2となるように遷移することが理解出来る。

【0041】このような状態遷移図に対応して、真理値表変換回路4からの真理値出力と、並べ換え回路7からの出力に基づき、論理式に変換する。

【0042】次いで、HDL変換回路11において、切り出し回路2からの非同期リセット生成信号が入力されているか否かを判断し（ステップS9）、非同期リセッ

ト生成信号が入力されている場合は、セクタ10から出力される、変換HDL選択信号91に対応するハードウェア変換情報に基づき非同期のリセット条件式から上記変換された論理式をハードウェア記述言語変換し、生成する（ステップS10）。

【0043】更に、状態名を含んだ真理値表（図9）からハードウェア記述言語を生成して（ステップS11）処理を終了する。この生成されたハードウェア記述言語は、後のCADシステムの運用のためにメモリ12に記憶蓄積される。

【0044】図11は、上記のハードウェア記述言語（本例ではVHDL）への変換の対象とした状態遷移回路の一例を示す図である。

【0045】図11の状態遷移回路は、フリップフロップ110、111を含み、更にこれらを制御する組み合わせ回路としてアンドゲート112～116、オアゲート117、118及びインバータ119～121を有している。

【0046】更に、図11の状態遷移回路の接続情報のみをリストしたネットリストが図12に示される。そして、このネットリストを先に図1乃至図10に基づき説明した本発明の実施例にしたがい、ハードウェア記述言語であるVHDLに変換した結果が図13及び図14に示される。尚、図14は、図13の記述に継続する記述である。

【0047】

【発明の効果】以上実施例にしたがい説明したように本発明により、既存の回路をハードウェア記述言語に変換することができ、これにより電気的特性やテクノロジーに依存しない機能シュミレーションあるいは、他のシステムへの移植性を高めることが可能となる。

【0048】更に、ハードウェア記述言語に変換することにより、ゲート数を少なくし、また処理スピードを高める目的にあったセルにマッピングが可能である。また、状態割り付け（状態のコード付け）の変更が可能となる。

【図面の簡単な説明】

【図1】本発明の実施例構成のブロック図である。

【図2】本発明の実施例の処理フローである。

【図3】FFのQN端子を含む状態遷移回路のブロック図である。

【図4】QN端子の省略を説明する図である。

【図5】図3のQN端子を削除した時の状態遷移回路のブロック図である。

【図6】組み合わせ回路30を真理値表に変換した結果を示す図である。

【図7】図6の現状態コード、次状態コードのdon't care（無視論理）を展開した真理値表である。

【図8】図7に現状態、次の状態に仮の状態名を設定した真理値表である。

【図9】現状態名でソートし、現状態名毎にまとめた真理値表である。

【図10】変換された状態遷移図である。

【図11】状態遷移回路の一例である。

【図12】図11の状態遷移回路に対応するネットリストの一例である。

【図13】変換されたハードウェア記述言語（VHDL）（その1）である。

【図14】変換されたハードウェア記述言語（VHDL）（その2）である。

【符号の説明】

1 ネットリストを記憶するメモリ

2 切り出し回路

* 3 出力負論理変換回路

4 真理値表変換回路

5 状態名変換テーブルを記憶するメモリ

6 状態名設定回路

7 並び換え回路

8 論理式変換回路

9 HDL変換情報

9 1 変換HDL選択信号

1 0 セレクタ

10 1 1 HDL変換回路

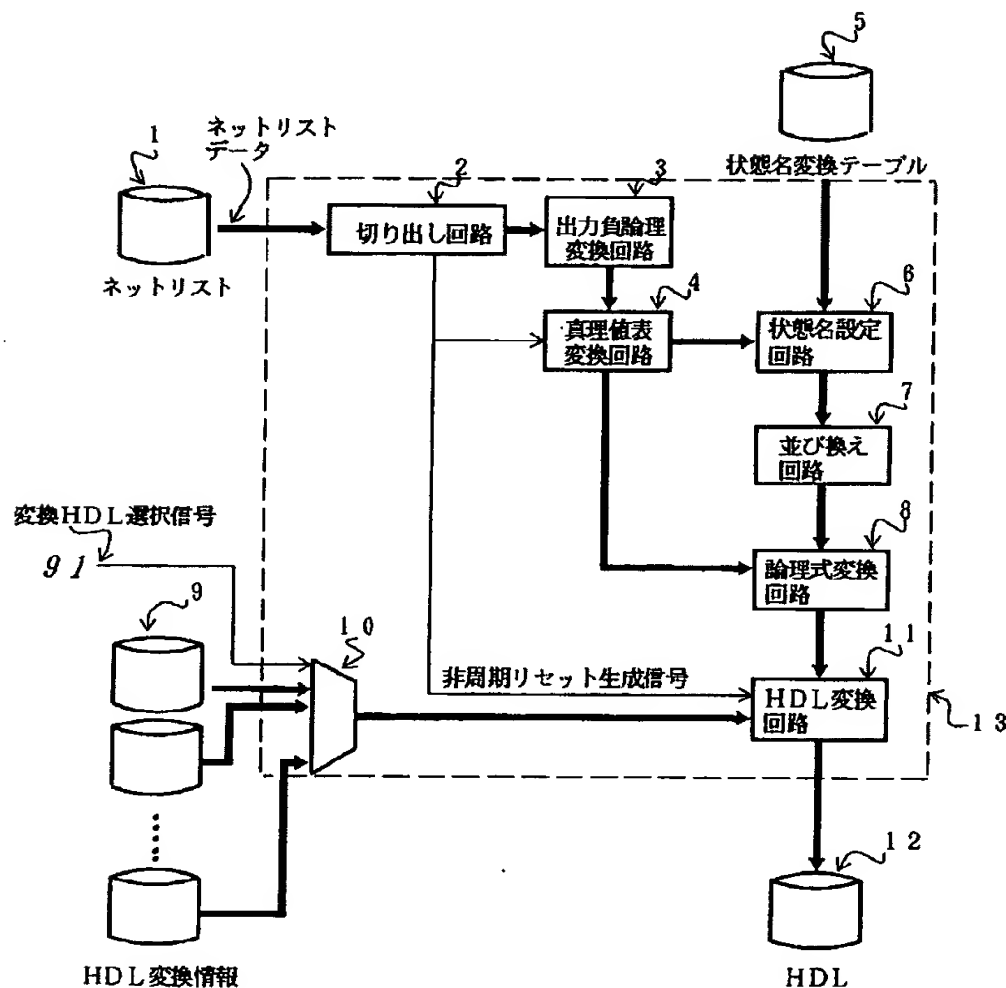
1 2 変換されたHDLを記憶するメモリ

1 3 変換処理を実行する演算処理装置

*

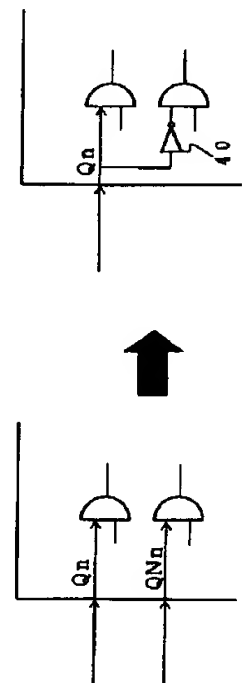
【図1】

実施例構成ブロック図



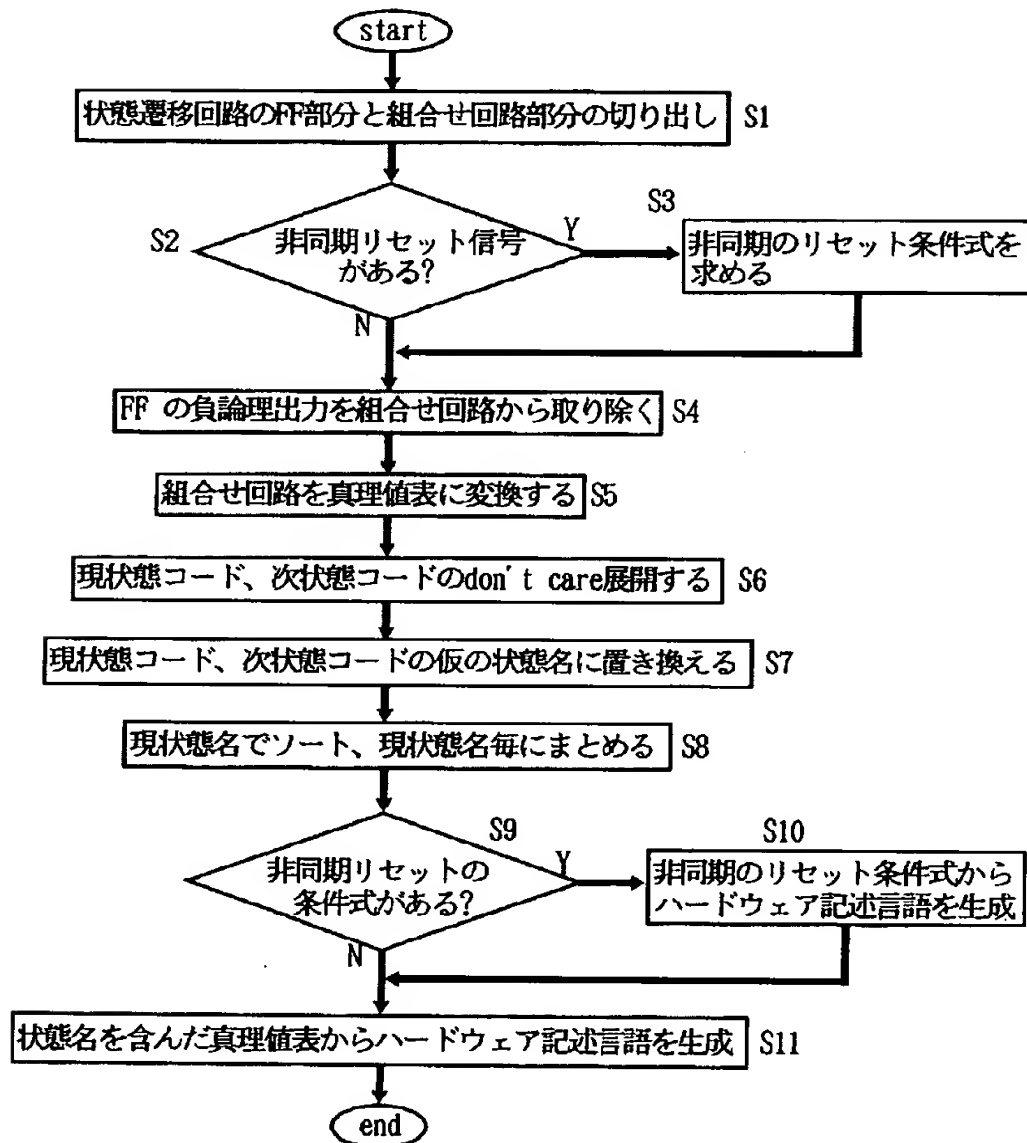
【図4】

Q n 端子の省略の説明図



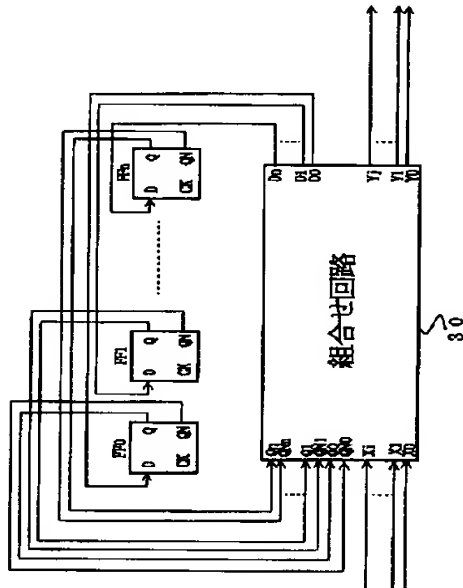
【図 2】

本発明実施例処理フロー



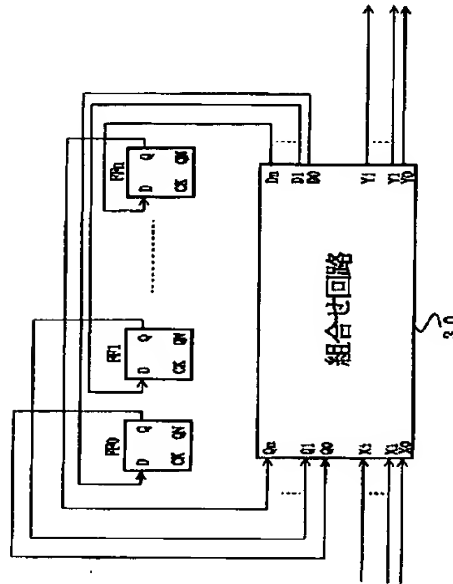
【図 3】

F FのQN端子を含む状態遷移回路ブロック図



【図 5】

図 3 のQN端子を削除した時の状態遷移回路のブロック図



【図 8】

図 7 に現状態、次の状態に仮の状態名を設定した真理値表

入力データ	現状態	次状態	出力データ
$X_i \dots X_1 X_0$			$Y_i \dots Y_1 Y_0$
0... 0 1	ST1	ST3	1... 1 1
⋮	⋮	⋮	⋮
0... 1 -	ST0	ST2	1... 1 0
0... 1 -	ST1	ST2	1... 1 0
⋮	⋮	⋮	⋮
1... 1 1	ST0	ST1	1... 1 0
⋮	⋮	⋮	⋮

【図 6】

組み合わせ回路 30 を真理値表に変換した結果

入力データ	現状態コード	次状態コード	出力データ
$X_i \dots X_1 X_0$	$Q_n \dots Q_1 Q_0$	$D_n \dots D_1 D_0$	$Y_i \dots Y_1 Y_0$
0... 0 1	0... 1 0	1... 0 0	1... 1 1
⋮	⋮	⋮	⋮
0... 1 -	0... - 0	1... 0 1	1... 1 0
⋮	⋮	⋮	⋮
1... 1 1	0... 0 0	0... 1 0	1... 1 0
⋮	⋮	⋮	⋮

【図 7】

図 6 の現状態コード、次状態コードの don't care を展開した真理値表

入力データ	現状態コード	次状態コード	出力データ
$X_i \dots X_1 X_0$	$Q_n \dots Q_1 Q_0$	$D_n \dots D_1 D_0$	$Y_i \dots Y_1 Y_0$
0... 0 1	0... 1 0	1... 0 0	1... 1 1
⋮	⋮	⋮	⋮
0... 1 -	0... 0 0	1... 0 1	1... 1 0
0... 1 -	0... 1 0	1... 0 1	1... 1 0
⋮	⋮	⋮	⋮
1... 1 1	0... 0 0	0... 1 0	1... 1 0
⋮	⋮	⋮	⋮

← A0

← A1
← A2

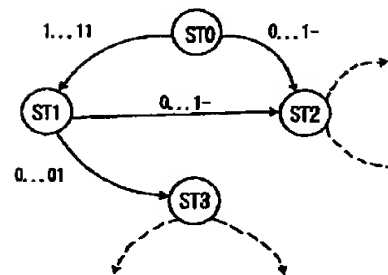
【図9】

現状態名でソートし、現状態名毎にまとめた真理値表

入力データ Xi...Xi X0	現状態	次状態	出力データ Yi...Y1 Y0
1...11 0...1-	ST0 ST0	ST1 ST2	1...10 1...10
⋮ ⋮	⋮ ⋮	⋮ ⋮	⋮ ⋮
0...1- 0...01	ST1 ST1	ST2 ST3	1...10 1...11
⋮ ⋮	⋮ ⋮	⋮ ⋮	⋮ ⋮

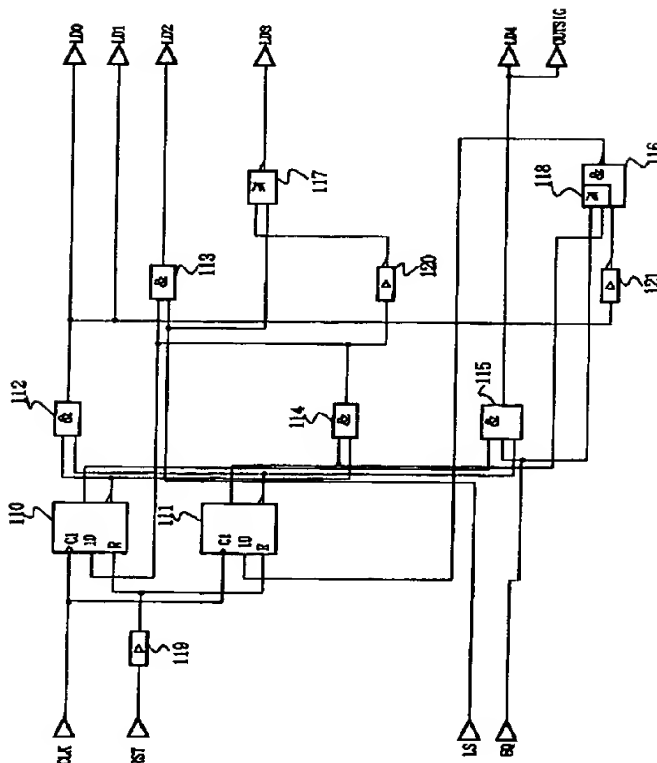
【図10】

変換された状態遷移図



【図11】

状態遷移回路の一例



【図12】

図11の状態遷移回路に対応する
ネットリストの一例

```

USER      : FUJITSU;
SYSTEM    : TEST;
REVISION  : 0001;
DATE      : 94/05/31;
DESIGNER  : FUJITSU;
NAME      : CNTL;
PURPOSE   : LOGSIL LAYOUT;
LEVEL     : UNKNOWN;
REVISION  : 0001;
DATA      : 94/05/31;
DESIGNER  : FUJITSU;
INPUTS    : CLK, RST, LS, BQ;
OUTPUTS   : LD0, LD1, LD2, LD3, LD4, OUTSIG;
TYPES     :
N2P : MOOEX, MOOPZ, MOOPY;
V2B : MOOGZ, MOOBZ, XXX001;
R2K : MOOCY;
N3P : MOOCX;
G23 : MOOEY;
P20 : XXX000, XXX100;
ENDTYPES;
NETS:
XXX001X : XXX001, X, XXX100, R, XXX000, R;
XXX000XQ : XXX000, XQ, MOOPZ, A2, MOOPY, A1;
MOOEZX : MOOEZ, X, MOOEY, B;
MOOGZX : MOOGZ, X, MOOEY, A1;
CLKZ : CLK, XXX000, CK, XXX100, CK;
RSTZ : RST, XXX001, A;
D2P_02 : XXX000, Q, MOOCX, A1;
P2D_02 : MOOPZ, X, MOOEY, A1, MOOGZ, A, XXX000, D;
D2P_12 : XXX100, Q, MOOEY, A2, MOOPZ, A1;
P2D_12 : MOOEY, X, XXX100, D;
LSZ : LS, MOOCX, A2, MOOEY, A2;
BQZ : BQ, MOOCX, A2, MOOEY, A1;
LD1Z : LD1, LD0, MOOPY, X, MOOEZ, A;
LD2Z : LD2, MOOCX, X;
LD3Z : LD3, MOOEY, X;
OUTSIGZ : OUTSIG, LD4, MOOCX, X;
NET0Z : XXX100, XQ, MOOCX, A3, MOOPY, A2;
ENDNETS;
ENDNAME;
ENDUSER;

```

【図13】

変換されたハードウェア記述言語 (VHDL) (その1)

```

entity QNTL is
  port(
    RST : in bit;
    EQ : in bit;
    LS : in bit;
    CLK : in bit;
    OUTSIG : out bit;
    LD4 : out bit;
    LD3 : out bit;
    LD2 : out bit;
    LD1 : out bit;
    LD0 : out bit
  );
end QNTL;

architecture state_transition of QNTL is
  type t_state is (ST0, ST1, ST2);
  signal state : t_state := ST0;
  signal next_state : t_state;
begin
  process(RST, CLK)
  begin
    if (RST = '1') then
      state <= ST0;
    elsif (CLK event and CLK = '1') then
      state <= next_state;
    end if;
  end process;

  process(EQ, LS, state)
  begin
    next_state <= state;

    case state is
      when ST0 =>
        OUTSIG <= '0';
        LD4 <= '0';
        LD3 <= '0';
        LD2 <= '0';
        LD1 <= '0';
        LD0 <= '0';

        next_state <= ST1;

```

【図14】

変換されたハードウェア記述言語 (VHDL) (その2)

```

        when ST1 =>
          OUTSIG <= '0';
          LD4 <= '0';
          LD3 <= '0';
          LD2 <= '0';
          LD1 <= '0';
          LD0 <= '0';

          if (LS = '1') then
            next_state <= ST2;
          elsif (not(LS) = '1') then
            next_state <= ST2;
          end if;

        when ST2 =>
          OUTSIG <= '0';
          LD4 <= '0';
          LD3 <= '0';
          LD2 <= '0';
          LD1 <= '0';
          LD0 <= '0';

          if (not(EQ) = '1') then
            next_state <= ST1;
          elsif (not(not(EQ)) = '1') then
            next_state <= ST0;
          end if;
        end case;
      end process;
    end state_transition;

```